Звіт до лабораторної роботи №1 з Дискретної математики

Виконавці: Віктор Мурин, Тарас Лисун

Посилання на репозиторій: https://github.com/hellcastter/discrete_mathematics_lab1.git

Задача: створити модуль на Python для читання та запису файлу відношення у формі матриці (у форматі txt) та базовими операціями над нею.

У папці discrete_mathematics_lab1 міститься 3 файли:

- **main.py** - модуль, що містить у собі функції для взаємодії з відношеннями
- **matrix.txt** - приклад матриці, яку необхідно використовувати при роботі з функціями
- **README.md** - опис задач для лабораторної роботи та перелік функцій, що є в модулі

Опис функцій для взаємодії з відношенням:

Основні завдання:

1. **read_matrix(file_name)**

   Відкриває текстовий файл (file_name), у якому повинна бути матриця у такому форматі:

   ```
   00010
   00010
   01000
   10000
   01011
   ```

   Після цього повертає матрицю користувача у вигляді списку списків, які містять цифри:

   ```
   [[0, 0, 0, 1, 0], [0, 0, 0, 1, 0], [0, 1, 0, 0, 0], [1, 0, 0, 0, 0], [0, 1, 0, 1, 1]]
   ```

   Подальші взаємодії з відношеннями будуть проводитись саме з результатом цієї функції (аргумент matrix).

2. **write_matrix(matrix, file_name)**

   Записує значення матриці(matrix) у текстовий файл (file_name) в такому вигляді:

   ```
   00010
   00010
   01000
   10000
   01011
   ```

3. **reflexive_relation(matrix)**

   Шукає рефлексивне замикання матриці. Простіше кажучи, замінює усі значення елементів головної діагоналі на 1.

   ```
   print('User matrix:', read_matrix('matrix.txt'))
   print('Reflexive relation of user matrix:', reflexive_relation(read_matrix('matrix.txt')))
   ```

   ```
   User matrix: [[0, 0, 0, 1, 0], [0, 0, 0, 1, 0], [0, 1, 0, 0, 0], [1, 0, 0, 0, 0], [0, 1, 0, 1, 1]]
   Reflexive relation of user matrix: [[1, 0, 0, 1, 0], [0, 1, 0, 1, 0], [0, 1, 1, 0, 0], [1, 0, 0, 1, 0], [0, 1, 0, 1, 1]]
   ```

4. **symmetrical_relation(matrix)**

   Шукає симетричне замикання матриці (для всіх пар (a, b) додає, якщо відсутня, пару (b, a)).

   ```
   print('User matrix:', read_matrix('matrix.txt'))
   print('Symmetrical relation of user matrix:', symmetrical_relation(read_matrix('matrix.txt')))
   ```

```
User matrix: [[0, 0, 0, 1, 0], [0, 0, 0, 1, 0], [0, 1, 0, 0, 0], [1, 0, 0, 0, 0], [0, 1, 0, 1, 1]]
Symmetrical relation of user matrix: [[0, 0, 0, 1, 0], [0, 0, 1, 1, 1], [0, 1, 0, 0, 0], [1, 1, 0, 0, 1], [0, 1, 0, 1, 1]]
```

5. **transitive_closure(matrix)**

   Шукає транзитивне замикання матриці за алгоритмом Уоршалла.

```
print('User matrix:', read_matrix('matrix.txt'))
print('Trasnitive relation of user matrix:', transitive_closure(read_matrix('matrix.txt')))
```

```
User matrix: [[0, 0, 0, 1, 0], [0, 0, 0, 1, 0], [0, 1, 0, 0, 0], [1, 0, 0, 0, 0], [0, 1, 0, 1, 1]]
Trasnitive relation of user matrix: [[1, 0, 0, 1, 0], [1, 0, 0, 1, 0], [1, 1, 0, 1, 0], [1, 0, 0, 1, 0], [1, 1, 0, 1, 1]]
```

6. **search_equivalence_classes(matrix)**

   Шукає класи еквівалентності на матриці. Повертає список класів, що є списками цифр

```
print('User matrix:', read_matrix('matrix.txt'))
print('Equivalence classes of user matrix:', search_equivalence_classes(read_matrix('matrix.txt')))
```

```
User matrix: [[0, 0, 0, 1, 0], [0, 0, 0, 1, 0], [0, 1, 0, 0, 0], [1, 0, 0, 0, 0], [0, 1, 0, 1, 1]]
Equivalence classes of user matrix: [[4], [3, 4, 5], [2], [1, 2, 5], [2, 4, 5]]
```

Додаткові завдання:

Завдання №5: **is_transitive(matrix)**

Перевіряє, чи є матриця транзитивним відношенням за допомогою наслідку з алгоритму Уоршалла.

```
print('User matrix:', read_matrix('matrix.txt'))
print('User matrix is transitive:', is_transitive(read_matrix('matrix.txt')))
```

```
User matrix: [[0, 0, 0, 1, 0], [0, 0, 0, 1, 0], [0, 1, 0, 0, 0], [1, 0, 0, 0, 0], [0, 1, 0, 1, 1]]
User matrix is transitive: False
```

```
User matrix: [[1, 1, 1, 1], [1, 1, 1, 1], [0, 0, 0, 1], [0, 0, 0, 0]]
User matrix is transitive: True
```

Завдання №6: **search_transitive_count(size)**

Перевіряє кількість усіх можливих транзитивних відношень на множині з size елементів. Оскільки не існує формули для підрахунку точної кількості транзитивних відношень, функція генерує та перевіряє всі можливі матриці.

```
print('Quantity of transitive relations', search_transitive_count(4))
```

```
Quantity of transitive relations 3994
```

У цій функції також міститься функція check_transitive(matrix), яка перевіряє матрицю на транзитивність та збільшує загальне число кількості транзитивних відношень на 1.

Допоміжні функції для search_transitive_count(size):

- **_apply_to_all_matrixes(size, callback)**

  size – кількість елементів у матриці

  Функція генерує усі можливі n-елементні матриці, але не виводить їх, задля збереження пам’яті та продуктивності. Ця функція містить у собі функцію generate_binary_matrixex(num, matrix, callback, index), яка, власне, і генерує усі n-елементні матриці.

Впродовж виконання лабораторної роботи ми використовували github задля полегшення роботи: https://github.com/hellcastter/discrete_mathematics_lab1.git